REMARKS

Claims 1-44 are pending. Claims 1, 3, 4, 17, 19, 20, 34, 35, 39, and 40 have been amended. Claims 1-44 remain in this application. No new matter has been entered.

The disclosure is objected for informalities. Accordingly, the disclosure has been amended per the Examiner's suggestions. Withdrawal of the objection to the disclosure is requested.

The abstract is objected for informalities. Accordingly, the abstract has been amended. Withdrawal of the objection to the abstract is requested.

The disclosure is objected for containing embedded hyperlinks. Accordingly, the disclosure has been amended per the Examiner's suggestions. Withdrawal of the objection to the disclosure is requested.

The drawings are objected for non-compliance with 37 CFR 1.84(p)(4). Accordingly, the disclosure has been amended. Withdrawal of the objection to the disclosure is requested.

Claims 4, 20, 35, and 40 stand rejected under 35 U.S.C. §112, second paragraph, as being indefinite. Accordingly, Claims 4, 20, 35, and 40 have been amended. Withdrawal of the rejection for indefiniteness is requested.

Claims 1-11, 14-27, and 30-44 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,951,698, issued to Chen et al. ("Chen"). Applicant traverses the rejection.

A claim is anticipated under 35 U.S.C. §102(e) only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. MPEP § 2131. The Chen reference fails to describe, either expressly or inherently, each and every claim element of, and therefore does not anticipate, Claims 1-11, 14-27, and 30-44.

Chen describes a system and method for detecting and removing macro viruses (Abstract), that includes a macro locating and decoding module, macro virus scanning module, macro treating module, virus information module, file correcting module, and a data buffer (Col. 2, lines 30-34). The macro locating and decoding module examines a target file and locates and decodes any macros into the data buffer (Col. 2, lines 34-45). The macro virus information module includes information that is used by the macro virus scanning module to detect both known and unknown macro viruses (Col. 2, lines 46-53) using comparison data that includes information that is used to detect combinations of suspect instructions in macros (Col. 2, line 64-Col. 3, line 1). The comparison data preferably includes several sets of instruction identifiers and various combinations of suspect instructions may be detected, including macro virus enablement or reproduction instructions (Col. 14, lines 51-60). The instruction identifiers are stored in the virus information module in a data table, which includes rows that correspond to several different instruction set identifiers and columns identifying an instruction set identifier, instruction identifier number, and text and corresponding hexadecimal representations of the binary code for the instruction identifiers (Col. 14, line 65-Col. 15, line 6).

In contrast, amended Claim 1 recites each macro virus definition data file defining macro virus attributes for known macro viruses that are each comprised of at least one macro. Claim 1 further recites the sets of the indices and the macro virus definition data files being organized into a *hierarchy* according to macro virus families based on a type of application to which the macro applies. Claim 1 further recites a parser parsing a suspect file into tokens comprising one of individual string constants and source code text and storing the tokens as suspect strings into a *hierarchical parse tree*. Claim 1 further recites a macro virus checker *traversing the hierarchical parse tree* to retrieve each suspect string (emphasis added). Support for the amendments can be found in the specification on page 7, line 19 through page 8, line 18.

In particular, Claim 1 recites organizing a hierarchy of indices sets and macro virus definition data files according to macro virus families based on application type, rather than a data table containing rows of instruction identifier sets, as described by Chen. Thus, per Claim 1, the macro virus family to which the suspect string belongs can be readily associated for each application type, such as word processor, spreadsheet, presentation, and generic applications types. In addition, Claim 1 recites parsing a suspect file into tokens that are stored as suspect strings in a hierarchical parse tree and traversing the hierarchical parse tree to retrieve each suspect string, whereas Chen describes storing information associated a decoded macro into a data buffer that is apparently traversed in a linear fashion. Per Claim 1, hierarchical parse tree traversal is efficient and allows ready comparison of individual string constants and source code text, including identification of macro virus definition files only partially containing a suspect string. Such limitations are neither taught nor suggested by Chen.

In contrast, amended Claim 17 recites each macro virus definition data file defining macro virus attributes for known macro viruses that are each comprised of at least one macro. Claim 17 further recites organizing the sets of the indices and the macro virus definition data files into a *hierarchy* according to macro virus families based on a type of application to which the macro applies. Claim 17 further recites parsing a suspect file into tokens comprising one of individual string constants and source code text and storing the tokens as suspect strings into a *hierarchical parse tree*. Claim 17 further recites *traversing the hierarchical parse tree* to retrieve each suspect string (emphasis added). Support for the amendments can be found in the specification on page 7, line 19 through page 8, line 18.

In particular, Claim 17 recites organizing a hierarchy of indices sets and macro virus definition data files according to macro virus families based on application type, rather than a data table containing rows of instruction identifier sets, as described by Chen. Thus, per Claim 17, the macro virus family to which the suspect string belongs can be readily associated for each application type, such as word processor, spreadsheet, presentation, and generic applications types. In addition, Claim 17 recites parsing a suspect file into tokens that are stored as suspect strings in a hierarchical parse tree and traversing the hierarchical parse tree to retrieve each suspect string, whereas Chen describes storing information associated a decoded macro into a data buffer that is apparently traversed in a linear fashion. Per Claim 17, hierarchical parse tree traversal is efficient and allows ready comparison of individual string constants and source code text, including identification of macro virus definition files only partially containing a suspect string. Such limitations are neither taught nor suggested by Chen.

In contrast, amended Claim 34 recites each macro virus definition data file defining macro virus attributes for known macro viruses that are each comprised of at least one macro. Claim 34 further recites a *hierarchy* organized according to a macro family to which each of the sets of the indices and the macro virus definition data files belong based on a type of application to which the macro applies. Claim 34 further recites a parser parsing a suspect file into tokens comprising one of individual string constants and source code text and storing the tokens as strings into *a hierarchical parse tree*. Claim 34 further recites a macro virus checker *traversing the hierarchical parse tree* to retrieve the strings (emphasis added). Support for the amendments can be found in the specification on page 7, line 19 through page 8, line 18

In particular, Claim 34 recites organizing a hierarchy of indices sets and macro virus definition data files according to macro virus families based on application type, rather than a data table containing rows of instruction identifier sets, as described by Chen. Thus, per Claim 34, the macro virus family to which the string belongs can be readily associated for each application type, such as word processor, spreadsheet, presentation, and generic applications types. In addition, Claim 34 recites parsing a suspect file into tokens that are stored as strings in a hierarchical parse tree and traversing the hierarchical parse tree to retrieve each string, whereas Chen describes storing information associated a decoded macro into a data buffer that is apparently traversed in a linear fashion. Per Claim 34, hierarchical parse tree traversal is efficient and allows ready comparison of individual string constants and source code text, including identification of macro virus definition files only partially containing a string. Such limitations are neither taught nor suggested by Chen.

In contrast, amended Claim 39 recites each macro virus definition data file defining macro virus attributes for known macro viruses that are each comprised of at least one macro. Claim 39 further recites organizing the sets of the indices and the macro virus definition data files into a *hierarchy* according to macro virus families based on a type of application to which the macro applies. Claim 39 further recites parsing a suspect file into tokens comprising one of individual string constants and source code text and storing the tokens as strings into *a hierarchical parse tree.* Claim 39 further recites *traversing the hierarchical parse tree* to retrieve the strings (emphasis added). Support for the amendments can be found in the specification on page 7, line 19 through page 8, line 18.

In particular, Claim 39 recites organizing a hierarchy of indices sets and macro virus definition data files according to macro virus families based on application type, rather than a data table containing rows of instruction identifier sets, as described by Chen. Thus, per Claim 39, the macro virus family to which the string belongs can be readily associated for each application type, such as word processor, spreadsheet, presentation, and generic applications types. In addition, Claim 39 recites parsing a suspect file into tokens that are stored as strings in a hierarchical parse tree and traversing the hierarchical parse tree to retrieve each string, whereas Chen describes storing information associated a decoded macro into a data buffer that is apparently traversed in a linear fashion. Per Claim 39, hierarchical parse tree traversal is efficient and allows ready comparison of individual string constants and source code text, including identification of macro virus definition files only partially containing a string. Such limitations are neither taught nor suggested by Chen.

Claims 2-11 and 14-16 are dependent on Claim 1 and are patentable for the above-stated reasons for independent Claim 1, and as further distinguished by the limitations recited therein. Claims 18-27 and 30-33 are dependent on Claim 17 and are patentable for the above-stated reasons for Claim 7, and as further distinguished by the limitations recited therein. Claims 35-38 are dependent on Claim 34 and are patentable for the above-stated reasons for Claim 34, and as further distinguished by the limitations recited therein. Claims 40-44 are dependent on Claim 39 and are patentable for the above-stated reasons for Claim 39, and as further distinguished by the limitations recited therein. As Chen fails to anticipate Claims 1-11, 14-27, and 30-44, withdrawal of the rejection for anticipation under 35 U.S.C. 102(e) is requested.

Claims 12-13 and 28-29 stand rejected under 35 U.S.C. §103(a) as being obvious over Chen. Applicant traverses the rejection.

To establish a *prima facie* case of obviousness: (1) there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or combine the reference teachings; (2) there must be a reasonable expectation of success; and (3) the combined references must teach or suggest all the claim limitations. MPEP §2143.

A *prima facie* case of obviousness has not been shown with respect to independent Claims 1 and 17. Claims 12-13 are dependent on Claim 1 and are patentable for the above-stated reasons, and as further distinguished by the limitations recited therein. Claims 28-29 are dependent on Claim 17 and are patentable for the above-stated reasons, and as further distinguished by the limitations recited therein. As a *prima facie* case of obviousness has not been shown, withdrawal of the rejection of Claims 12-13 and 28-29 for obviousness under 35 U.S.C. §103(a) is requested.

The prior art made of record and not relied upon has been reviewed by the applicant and is considered to be no more pertinent than the prior art references already applied.

Claims 1-44 are believed to be in a condition for allowance. Entry of the foregoing amendment is requested and a Notice of Allowance is earnestly solicited. Please contact the undersigned at (206) 381-3900 regarding any questions or concerns associated with the present matter.

Respectfully submitted,

Dated: December 10, 2004

By: [illegible]
Patrick J.S. Inouye, Esq.
Reg. No. 40,297

Law Offices of Patrick J.S. Inouye
810 Third Ave, Suite 258
Seattle, WA 98104

Telephone: (206) 381-3900
Facsimile: (206) 381-3999